A new space subdivision method for ray tracing CSG modelled scenes

Thierry Priol, Bruno Arnaldi, Kadi Bouatouch

INRIA

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 613

A NEW SPACE SUBDIVISION METHOD FOR RAY TRACING CSG MODELLED SCENES

Thierry PRIOL
Bruno ARNALDI
Kadi BOUATOUCH

Février 1987

IRISA INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

Une Nouvelle Méthode de Subdivision Spatiale pour le Lancer de Rayon Utilisant la Modélisation de Type CSG

A New Space Subdivision Method For Ray Tracing CSG Modelled Scenes

Thierry Priol
Bruno Arnaldi
Kadi Bouatouch

Publication Interne n° 337 - Janvier 1987 - 24 pages

Résumé : Nous présentons dans ce rapport un nouvel algorithme de lancer de rayon par subdivision de l'espace tout en utilisant la modélisation de type CSG. L'espace est découpé de façon irrégulière afin d'englober au mieux les primitives géométriques. Deux étapes sont nécessaires : un premier découpage est effectué sur un plan de projection puis l'extension en profondeur de ce découpage crée un ensemble de régions 3D. L'arbre CSG initial est distribué dans chacune de ces régions. Lors du lancer de rayon, la recherche de la prochaine région dans la direction du rayon est effectuée en utilisant des informations locales à chaque région. Nous détaillons une amélioration appelée "Boîte aux lettres" pouvant être utilisée dans les algorithmes de lancer de rayon mettant en oeuvre la subdivision de l'espace. Nous comparons dans la fin de ce rapport notre nouvel algorithme avec celui proposé par Roth.

Abstract : A new algorithm for space tracing with CSG modelled scenes is presented. Space is divided in an irregular fashion to fit the objects as closely as possible. For this reason, primitive minimal bounding boxes are computed. Space subdivision is achieved in two steps: partitionning in projection plane and depth partitionning. A set of 3D regions named cells are then created. Boolean CSG tree is distributed into the cell structure to form in each cell the minimal boolean CSG tree using the relevant primitives. The searching process for the "next cell" along the ray path is performed by using a local data structure associated with each cell. The goal of this structure is to link the cells together. An improvement, named "mailbox", for all space tracing algorithms is detailed. Results are presented for two scenes to compare this new algorithm with Roth's algorithm.

1. Introduction

Ray-casting (or ray tracing) is the most powerful technique for realistic image computation. The main interest of this method lies in the fact that a sophisticated illumination model can take into account specular reflection, transparency and shadow effects [Torrance 67, Phong 75, Blinn 77, Whitted 80, Cook 82, Hall 83].

Ray tracing simulates the operation of a camera, following light rays in reverse order [Goldstein 71 and 79, Roth 82, Kajiya 83]. In fact, the essential work consists in evaluating ray object intersections. The ray tracing process for a given ray is to compute all intersections between the ray and the primitives on its path to find the closest object.

Unfortunately, the basic ray tracing algorithm [Roth 82] is extremely time-consuming; this is mainly due to the large number of intersection calculations between elementary primitives and rays. To increase ray tracing efficiency, ray-object intersections procedures could be optimized. This goal can only be achieved by the use of special purpose hardware. Another issue is to improve data structure management to allow fast access to geometrical information. To overcome the large number of intersection calculations, several attemps have been made.

A first method consists in taking into account the property of *image coherency* [Bronsvoort 84].The intensity is computed for a group of pixels named macro pixel. The evaluation of each pixel intensity is then performed by extrapolation of the neighbouring macro pixels intensities. However, some slices of the image can disappear due to the ray traced sampling which is in opposition with actual tendency of works in ray tracing.

A second technique uses the *object coherence* [Coquillart 85] by projecting the object bounding volumes on a plane and building a graph for adjacency and superposition relations. However, when a great number of objects are situated at the same location on the plane, all of them contain a potential intersection point requiring effective intersection calculation. therefore spatial coherency is not fully used.

The third method to increase ray-tracing efficiency is based on *space subdivision*. Two importants points are to be considered: firstly the subdivision algorithm with its associated data structure, secondly data organization to provide access to geometrical information. The space where the three dimensional scene lies is divided into sub-spaces. The global data structure is distributed among these sub-spaces. In this case, the process for a given ray is to compute the intersections within each local data structure, until a real intersection point is found. Such a method globally reduces the number of intersection calculations for each ray.

The technique chosen for space subdivision is of great importance, as it has consequences on the storage requirement needed and the effectiveness of the searching process. Generally, space is divided recursively to generate a regular partition of the space.

In [Mantyla 83, Tamminen 84], the structure, used for modelling a solid in boundary representation, consists in two parts :

- A data structure : local geometric information about scene.
- A directory : array providing access by location to data cells.

The use of a directory makes it possible to move from one local geometric structure to the next one on the ray path.

Glassner uses the *octree* (octal tree) structure [Glassner 84], to provide a means of representing both the geometric information and the access to them. The leaves of the octree are the sub-spaces and the nodes contain the information necessary to select the path from a given cell to another one by traversing the octree from the root to a leaf.

The *binary space partitioning* (BSP) [Kaplan 85] is a binary tree, where each node contains the identification of a slicing plane and the leaves describe 3D regions with geometric information. As in the previously mentioned technique, the tree must be traversed in order to select the next cell on the ray path.

All these subdivision methods work with an arbitrary subdivision scheme. At each step of the subdivision, the space is divided by slicing planes wherever the objects are localized. In spite of adaptability in the space division by the use of a stop criterion in the recursive algorithm, these techniques produce a set of parallelepipeds where it is possible to find a large box containing a small object.

An important point in image synthesis concerns the object modelling. Different techniques are used to represent geometrical information. The simplest model consists in approximating the object surface by a set of planar polygonal facets. This model can be refined by the use of parametric polynomial surface like biquadratic or bicubic surfaces. In this case, ray-object intersection is more difficult to compute. The two previous models belong to the boundary representation type of model. Another choice is to consider volumic information. Some elementary primitives, like sphere, parallelepiped, etc.., can be used to represent the parts of an object. There are some limitations to the complexity of the objects which can be modeled by the use of these primitives. So a hierarchical model, which allows to perform boolean operations on volumes, gives a better way to represent complex objects.

This model is the well known CSG (Constructive Solid Geometry) [Requicha 77]. In CSG, an object is modeled as a binary tree whose nodes represent set operations (union, intersection, difference), and whose leaves are elementary solid primitives (sphere, parallelepiped, cylinder,etc...).

In this paper we propose a new method for space subdivision that fits closely to the spatial location of the objects in the scene and uses the CSG model. On the one hand, our subdivision is directly controlled by geometric information about elementary primitives. On the other hand, this technique provides a local search for the next cell on the ray path. An overview of our method is given in the next section.

2. Overview of the method

This new algorithm partitions the space with parallelepipeds, named cells, which fit as closely as possible the solid primitives used to build the scene. For this reason, we compute for each primitive its minimal bounding box. These minimal bounding boxes are used to perform an irregular partition of the space . Two steps are necessary to accomplish this task: the first one is a 2D subdivision and the second one is its extension to 3D. The result is a collection of cells which are linked together to speed up the "next cell" searching when space tracing. Each cell is associated with a minimal subtree which is derived from a "scene CSG tree".

Section three details the subdivision process. The data structure used to determine the next cell on the ray path is described in section four. Section five gives the space subdivision algorithm. Section six gives techniques for distributing the CSG tree in each cell and section seven describes the space-tracing algorithm using this data structure. Finally, an optimisation of space tracing based on "mailboxes" use, and results are presented.

3. Space subdivision

The aim of our space subdivision scheme is to create a set of cells which contains, or not, elementary primitives of the scene. The cells form a partition of the 3D space bounding the whole scene. In this section we describe the way to achieve the primitive minimal bounding boxes according to the CSG tree operators. Next we present the spatial subdivision in two logical steps: the first one is performed on a particular plane and the second is its extension to 3D.

3.1 Minimal bounding boxes

The goal is to compute the bounding boxes only for the part of the primitive which is preserved after applying successive boolean operators. The initial box bounds the whole primitive, however in some cases only a part of this box is effectively used.

Figure 1 shows two primitives A and B combined by an intersection operator. The effective minimal bounding box of A is included in the initial A bounding box : the boolean intersection of A and B bounding boxes. This example can be generalized for all the CSG tree. In fact, two steps are used to achieve this transformation: firstly boolean operations are performed on the primitive bounding boxes, depending on the CSG tree operators to compute node bounding boxes, secondly primitive minimal bounding boxes are computed for a given primitive as the intersection of all the bounding boxes when traversing the tree from the root down to the considered primitive. Algorithm 1 provides the minimal bounding boxes when it is applied to the root of the binary tree and its bounding box.

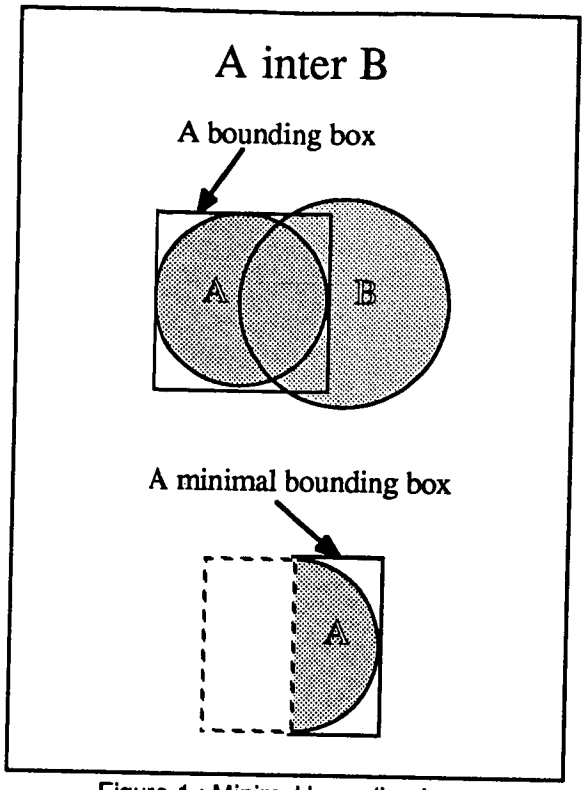

Figure 1 : Minimal bounding box.

```
procedure min_bounding_box(node, mini_bound)
var new_bound          { local minimal bounding box }
begin
   { intersection between minimal and current box }
   inters_bound (mini_bound, node.bound, new_bound)
   if leaf (node)
        then  { leaf of the CSG tree }
              affec_bound (node, new_bound)
        else  { node of the CSG tree }
              min_bounding_box(node.right_branch,new_bound)
              min_bounding_box(node.left_branch,new_bound)
   endif
end
```

Algorithm 1 : Bounding box transformation

In algorithm 1, inters_bound (b1,b2,b3) computes in b3 the intersection between the two bounding boxes b1 and b2, leaf (n) is true if n is a leaf and affec_bound (n,b) associates the bounding box b with the node n.

Once the primitive minimal bounding boxes are computed, subdivision process may begin. It is performed in two logical steps (wich are actually mixed in our implementation): the first one is made on screen plane and the second one along depth axis (z axis).

3.2. Partitionning in projection plane

Let us consider the perspective projection of the primitive minimal bounding boxes of the scene on the infinite projection plane, as shown in figure 2.

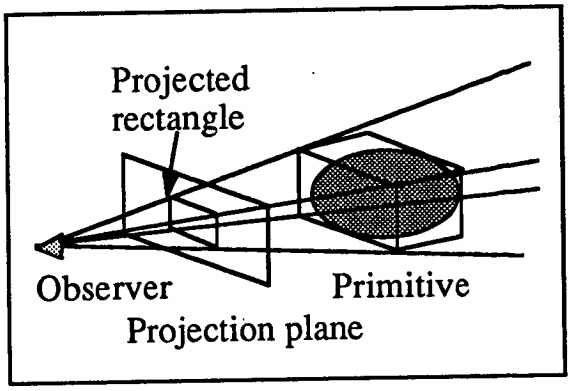

Figure 2 : Minimal bounding box perspective projection.

This projection generates a set of rectangles on the screen. Each rectangle is then decomposed into its four segments, used to perform a binary space partitioning [Fuchs 80]. The way to carry out the BSP is to choose one segment among the available segments list. The line, containing this segment, separates the plane into two regions. According to the segments location, the segments list is broken into two other lists, each one is associated with a region. The line extension of the chosen segment, may intersect other segments. In this case they are splitted and put in the two new lists. The same process is applied Recursively to these regions, until the associated lists become empty (Figure 3). Terminal boxes, containing or not a subset of primitives, are then created.

Different strategies can be applied for the segment choice. An arbitrary choice leads to a non-optimal screen partition, mainly due to the number of excessive splitting of line segments. Another approach consists in choosing the segment that intersects the minimal number of other segments to reduce the slicing tree. This technique gives good results but is time consuming. A better way, for the futur developments of this method, is to give to the segments a tree structure. Thereby, it becomes possible to search the optimal line segment in a sub-set of the segment list. This structure would provide an efficient technique to control the subdivision refinement especialy for very complex scenes.

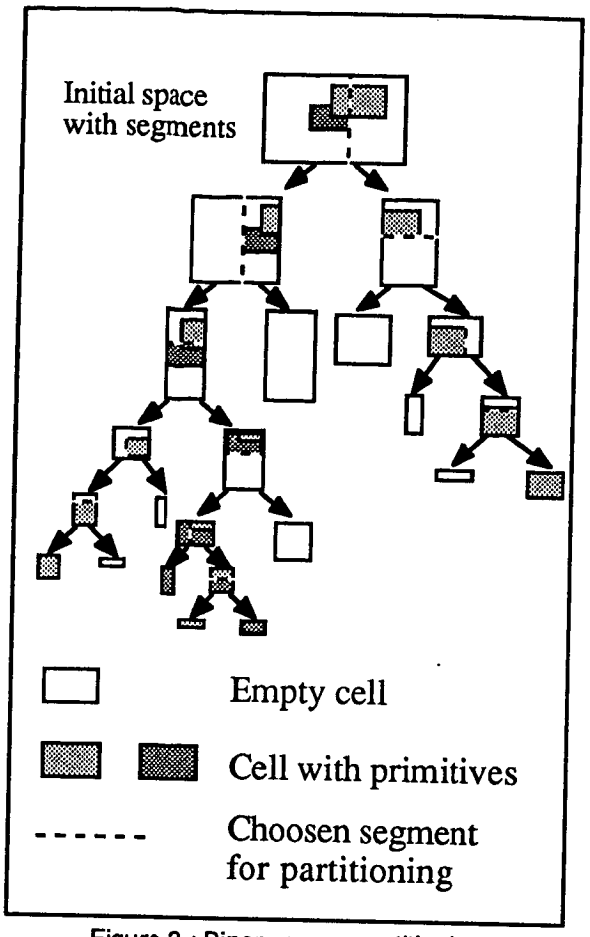

Figure 3 : Binary space partitioning.

This BSP process results in a 2D partition on screen plane, whose extension along depth axis gives 3D cells, called super cells (figure 4). Each of them may be empty or may contain some primitive objects. Non empty super cell projection defines a "Pixel area" , in the sense that they are the only screen areas that are ray-traced.

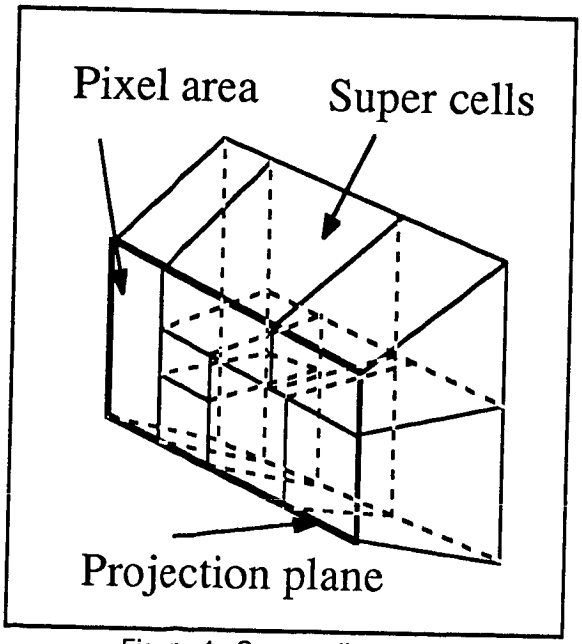

Figure 4 : Super cells structure.

To take advantage of the spatial coherence of the scene, a spatial subdivision is accomplished on the super cell structure during the second step.

3.3. Depth partitionning

The goal of this step is to refine the subdivision in the third dimension for each super cell which contains some primitives, distributed along the Oz axis. Let us assume that Ox and Oy axes are supported by the screen plane, where O is the coordinate system origin. A second application of the minimal bounding boxes allows the space partitionning along the Oz axis. The depth partitionning of each super cell generates a set of new cells.

Let Z_i be a super cell containing the primitives P_1 and P_2 with the respective minimal bounding values $Zmin_1$, $Zmax_1$, $Zmin_2$, $Zmax_2$. The subdivision process gives , in this example, five distinct cells from one super cell (Figure 5) .

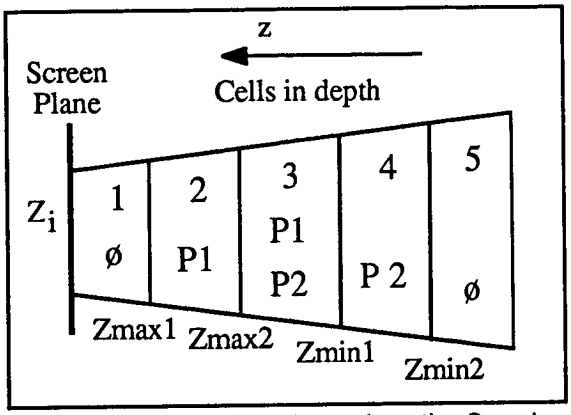

Figure 5 : Depth subdivision along the Oz axis.

As shown in this section, the cell structure represents an irregular partition of the initial space. The next section describes a data structure that allows fast space-tracing.

4. Cell connexity

A straightforward representation of this partition is not a sufficient data structure for space-tracing algorithm. The crucial problem consists in searching "next cell" along ray path. To accomplish this task, local information are added to each cell, named *connexity information*, to provide a means for localizing its neighbouring cells. In a first approach, we used lists to represent these information: one list per cell face exept for front and back faces. For a given face, the associated list contains pointers to all neighbouring cells which are adjacent to this face. The drawbacks of this structure are :

- dynamical size to represent connexity.
- linear search in a list.
- globally heavy size structure for all cells.
- redondant information to ensure connexity.

An alternative representation is derived from the corner stitching method of Ousterhout [Ousterhout 84]. This technique is used to express 2D VLSI layout. For one cell, we use a fixed set of pointers associated with the cell corners. The meaning of one pointer, is to link cells by their corners. The 3D extension is made from the particular connexity relation between cells on the Oz axis (Figure 4). Only one pointer is necessary to cover all cells lying in the direction of this axis. Figure 6 shows the pointer structure necessary for one cell.

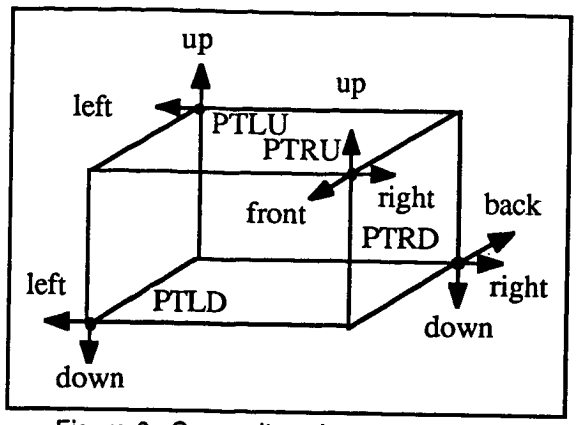

Figure 6 : Connexity using corner pointers.

PTLU, PTRU, PTRD, PTLD name each corner used in our method and up, down, front, back, left and right are the names of the associated pointers according to the direction of the adjacency relation.

With this technique, we can follow the path of a ray across the cell structure. Figure 7 shows a 2D projection of this structure and the path of a ray using pointers. Arrows indicate pointers effectively in use, and straight lines the other pointers.

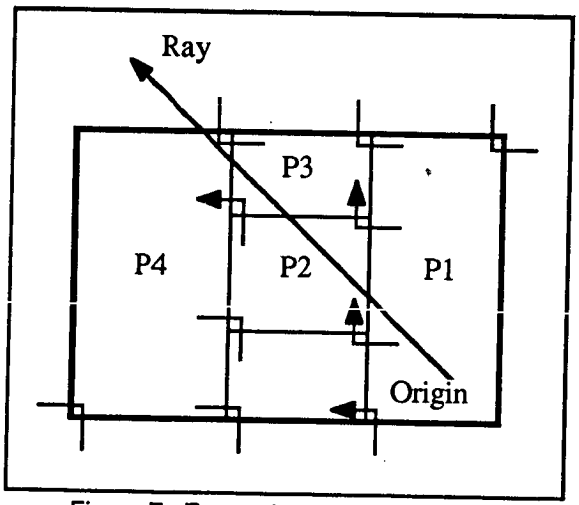

Figure 7 : Ray path across the structure.

In this example, P1 contains the origin of the ray. The use of the left down corner associated pointer in the left direction gives a cell which is not on the ray path. But in this cell the right up corner pointer in the up direction gives an access to the P2 cell and successively we can reach the cells P3 and P4.

5. Subdivision algorithm

To speed up the subdivision step, when a termination box is created during the 2D subdivision, depth subdivion is directly applied to its correspondant super cell to obtain the cells, and connexity information about these cells are updated. We produced an efficient recursive algorithm to implement these techniques. For this, we make use of the fact that :

When a terminal super cell is found during the recursive 2D subdivision algorithm, all cells on the top and on the right of the current super cell have already been treated.

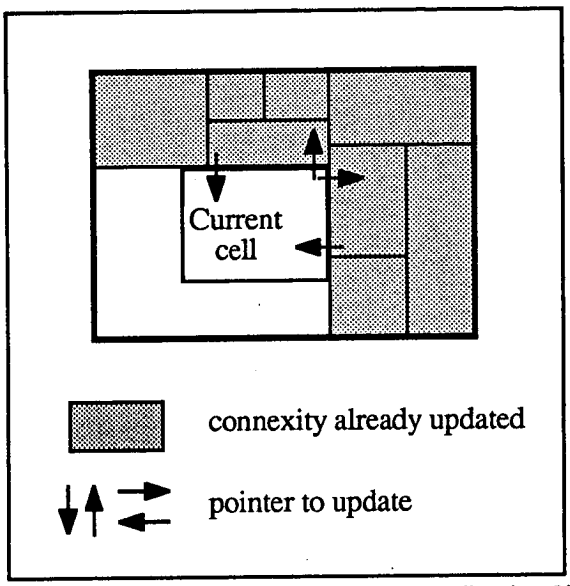

Figure 8 : pointers updating (2D generalized to 3D).

The outcome of this remark is to update only necessary pointers : right and up pointers of the cells, resulting from the current super cell, and all neighbouring cells pointers which must point to the current cells (Figure 8). The following algorithm shows how to obtain the complete structure.

Algorithm 2 : Space subdivision

```
procedure bspt ( var  r, l1, l2 , nl1, nl2 : ptr_cell);
var r1, r2, ltrav : ptr_cell ;
    select_seg : integer ;
begin
  if term_cell ( r )
      then  begin
              (*  r : terminal cell *)
              depth_subdi ( r ) ;
              if nl1 = nil then  nl1 := r ;
              if nl2 = nil then  nl2 := r ;
              connexity ;
            end
      else  begin
              (* division of r in two parts *)
              choice_segment (r, select_seg ) ;
              partition ( r, select_seg, r1 ,r2 ) ;
              ltrav := nil ;
```

```
            if horizontal (select_seg)
               then   begin
                        bspt ( r1, l1, l2, ltrav, nl2 ) ;
                        bspt ( r2, ltrav, l2, nl1, nl2 ) ;
                      end
               else   begin
                        bspt ( r1, l1, l2, nl1, ltrav ) ;
                        bspt ( r2, l1, ltrav, nl1, nl2 ) ;
                      end;
         end;
end; (* bspt *)
```

where

term_cell (r)	:	true if *r* is a teminal cell ,else false.
depth_subdi (r)	:	depth subdivion for the *r* cell.
connexity	:	updates cell pointers.
choice_segment (r, s)	:	chooses a segment *s* in the *r* associate list.
partition(r, s, r1, r2)	:	distributes list segments from *r* to *r1* and *r2*.
horizontal (s)	:	true if *s* is an horizontal segment else false.
r	:	current cell to be treated, wich may be terminal or not terminal.
l1 (resp. l2)	:	pointer giving the r^PTRU.up (resp. PTRU.right) value; its value can be modified in *connexity* to ensure the subdivision of the current cell.
nl1 (resp. nl2)	:	is a result of *bspt* call to prepare future recursive call; nl1 (resp. nl2) can be use as l1 (resp. l2) for these calls.

6. Distributing CSG tree

The purpose of this section is to describe how the whole CSG tree is distributed among the cells created by the subdivision step. In CSG modelling, each leaf is a simple solid primitive like sphere, cylinder,cone,etc.. and the operators (union,difference,intersection) represent the nodes of the tree (Figure 9).

We use a null operator whose effect is similar to union for objects that do not intersect. This new operator allows to speed up intersection list computation.

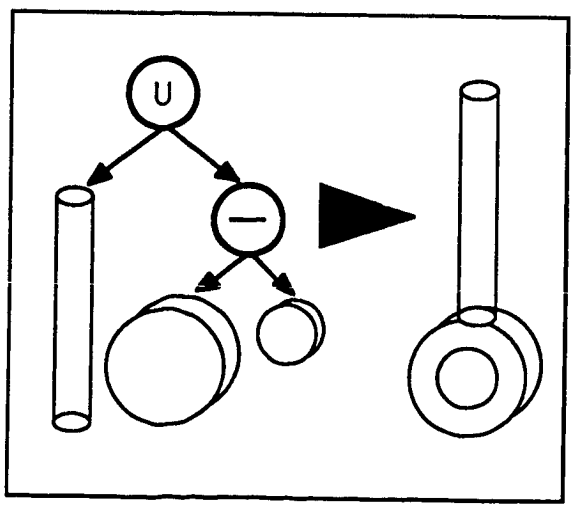

Figure 9 : CSG tree for modelling an object.

Distributing the whole CSG tree consists in deriving the relevant part of data structure for each cell. This is performed in two steps : firstly the whole CSG is restricted in a super cell giving a sub-CSG tree, secondarily this sub-CSG tree is restricted in its turn in each cell of this super cell provided by depth subdivision process described before. To make this possible, we use simple rules as follows :

Let be :

- *proj(node)* the projection of a node in the original CSG tree.
- P_i is a primitive if the node is a leaf.
- A, B are the projections of the right and left sub-trees of node.
- *op* is an operator.

Two cases may happen :

1) the node is a leaf :

if the primitive P_i, associated with the leaf, belongs to the cell
then the projection of the node is P_i
otherwise the projection is empty.

2) the node is not a leaf :

If A and B are not empty then *proj(node)* =A *op* B.
If A and B are empty then *proj(node)* =empty.

if A is empty and B is a tree then :
if op = ass then *proj(node)* =B
if op = union then *proj(node)* =B
if op = sub then *proj(node)* =empty
if op = inter then *proj(node)* =empty

if A is a tree and B is empty then :
if op = ass then *proj(node)* =A
if op = union then *proj(node)* =A
if op = sub then *proj(node)* =A
if op = inter then *proj(node)* =empty

All the primitives included in a cell are stored in an associated data structure. Then we take the original CSG tree and mark the leaves which belong to the cell structure. The next step consists in running the following recursive procedure written in pseudo-pascal :

```
procedure MakeSubTree(n:node;var ar:tree);
var ar1,ar2:tree;
    op:operator;
begin
  op:=n.op;      { operator of the node n }
  if op=fant     { the node represent a pointer to a leaf }
     then begin
            { it's a leaf }
            if primitive is contained in cell
               then begin
                      ar.op:=fant;
                      ar.ptrprimit:=n.ptrprimit;
                    end
               else ar:=empty
     else begin
            { it's a node }
            MakeSubTree(n.leftson,ar1);
            MakeSubTree(n.rightson,ar2);
            if (ar1=empty) and (ar2=empty)
               then ar:=empty
               else if (ar1<>empty) and (ar2<>empty)
                       then begin
                              ar.op:=op;
                              ar.leftson:=ar1;
                              ar.rightson:=ar2;
                            end
                    else if (ar1<>empty)
                            then if op<>inter
                                    then ar:=ar1
                                    else ar:=empty
                            else if (op=union)
                                          or (op=ass)
                                    then ar:=ar2
                                    else ar:=empty;
          end;
end;
```

Algorithm 3 : Create sub trees.

This operation creates a small sub-tree in each cell and the maximum height of a sub-tree depends on the number of primitives which overlap in the cell.

7. Ray tracer in a space subdivided into cells

As shown in figure 4 and 5, the 3D extension of a pixel area gives a non symmetric pyramidal polyhedron. With such a representation, the computation of the next cell from the previous one along the ray path, is time consuming (figure 10a). This may be done by computing the intersection beetwen a straight line (ray) and the planes of the cell faces. However, some of the faces are not parallel to the axes and cause more floating-point operations. To speed up the computation time, the geometrical information about cells are described using the well-known projection transformation [Newman 79, Foley 82], which has the following properties :

- Perspective projection becomes parallel.
- The x and y values are the usual perspective projection values.
- The z value is transformed to ensure that a plane in the absolute coordinate system gives a plane in the screen coordinate system. The z value order is preserved in this system.

Using such a system, the intersection of a ray and a cell boundary is greatly simplified, as shown in figure 10. A ray is expressed at once in the absolute coordinate system to compute object intersection points, and in the screen coordinate system to follow ray path throught the cell structure.

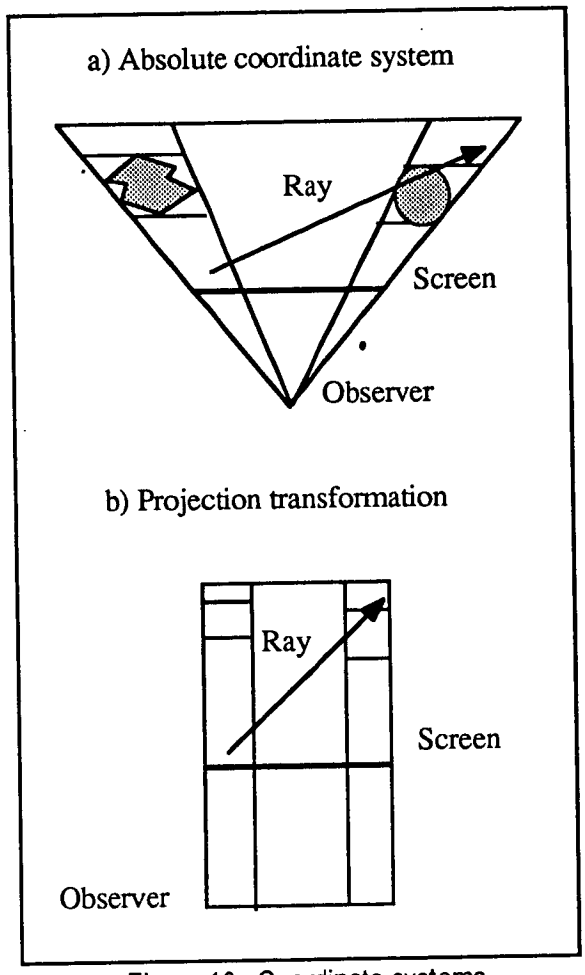

Figure 10 : Coordinate systems.

Once the preprocessing step is completed, the algorithm casts "primary ray" emanating from the viewpoint and passing through each pixel of the pixel areas ; these pixel areas were previously stored in an array by the preprocessing step. When a ray enters a cell, only objects belonging to this cell are intersected. If no intersection is found, a next cell in the direction of the ray is determined and the same process is applied until either an object is intersected or the ray leaves the scene. The search of the next cell on the "primary ray" path is fast because there is only one neighbouring cell on a back or front side of a cell (Figure 11).

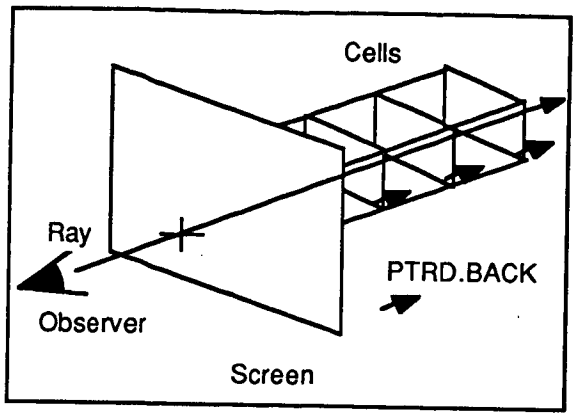

Figure 11 : Primary ray path.

Other rays (secondary rays, light rays) will likely intersect the east, west, south, or north face of a cell. Then, the search of the next cell on the ray path is more expensive because many cells may be connected to the cell containing the origin of this ray (Figure 12).

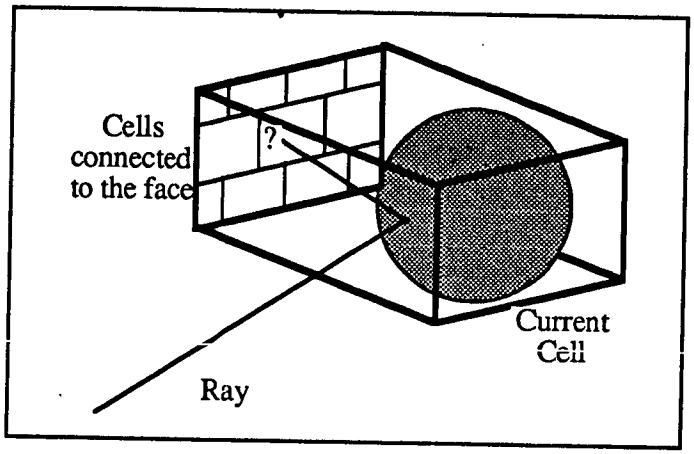

Figure 12 : Cells connected to a face.

For this operation, we use pointers associated with the four corners of the cell. Let us consider the case of Figure 13. The bold trace represents the west side of the cell 'rcour' intersected by the ray at the point represented by the cross. With the value of the pointer 'rcour^.ptld.left', cell connected to it is known. Then the pointer 'ptru.up' is used repeatedly to climb up to a cell which lower and upper bound contains the Y value of the intersection point. After that, the pointer named 'ptrd.back' is used until the upper bound of the cell is greater than the Z value of the intersection point.

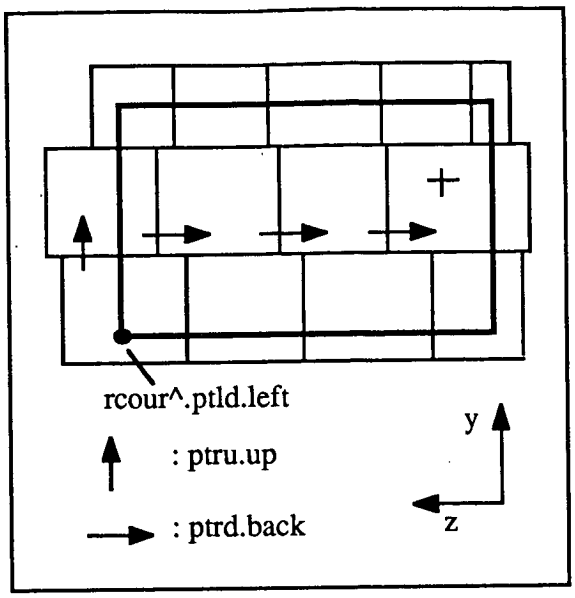

Figure 13 : Search of the next cell in ray path.

As an alternative, the pointer 'rcour.ptlu.left' could be used but no effort has been made to choose the pointer which minimizes search time. This improvement has got to be proved.

8. Mailbox

A well known drawback of space subdivision techniques is the possibility of having the same object in many cells thus implying repeated intersections with the same objects (Figure 14). To avoid this problem, we associate a "mailbox" to each primitive; each ray is numbered in a unique manner. The goal of the new structure is to store the latest intersection list between the primitive and a ray as well as the ray number. Thereby, before computing an intersection with a primitive, the algorithm examines the contents of the associated "mailbox". If the ray number stored is the same as the one of the current ray, the previous computation is available in the "mailbox", otherwise a new computation is necessary, which will be stored in its turn. This mechanism works for any kind of ray and increases ray-tracing efficiency.

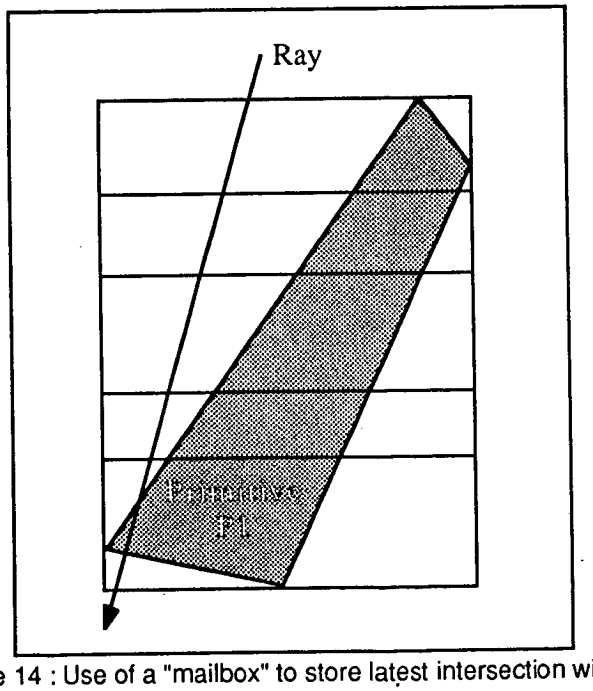

Figure 14 : Use of a "mailbox" to store latest intersection with a ray.

9. Results

The coding of the algorithm, described above, has been written in pascal language. Times are calculated on a Ridge - SPS9 running under Unix System V. The performance of this computer is approximatively the same as VAX 780. No effort has been made to optimize critical section of code, like intersection procedure. Thus the results obtained with the algorithms described above have been compared with the basic algorithm from S. Roth. The two programs run with Whitted and Phong illumination model. The light may illuminate either in all directions or in vector direction. All pictures are created using LGRC [Bouatouch 86] which is a language for geometric solid modeling, developed at IRISA.

Figure 15 represents the interior of a house. There are four lights to illuminate the scene and the maximum depth of ray tracing is two, which allows specular reflection in the mirror hanged on the wall. The pictures are calculated in a 1024 x 1024 resolution. Statistical tests are computed in 256 x 256 resolution. Figure 16 shows the efficiency of our algorithm on CSG model. The jagged ball is modelled using substraction and intersection operators with sphere and parallepiped primitives. Three lights are used and the maximum depth of ray tracing is two, which allows complex specular effects on the cube and on the ball. Resolution for this picture is the same as figure 16.

Table 1 shows the statistical results for the two images named "home" and "balls". The gain factor between the two methods can be increased by performing statistical tests on pictures which have a more important resolution. We present also the efficiency of the "mailbox" used in our algorithm. Mailboxes seem very efficient since they contain nearly 80 percent of the intersection results. Therefore, a lot of intersection calculations are saved.

	Home	Balls
NUMBER OF SOLID PRIMITIVES	71	63
NUMBER OF RAY TRACED	449,663	436,197
NUMBER OF INTERSECTIONS (Roth)	3,685,407	2,067,072
AVERAGE INTERSECTIONS PER RAY (Roth)	8.20	4.74
NUMBER OF INTERSECTIONS (New Method)	1,586,798	1,184,219
AVERAGE INTERSECTIONS PER RAY (New Method)	3.53	2.71
CPU TIME (HR:MIN.SS) (Roth)	11:40.55	37:40.43
CPU TIME (New Method) - preprocessing	00:01.50	00:07.04
- synthesis	01:17.40	01:25.02
GAIN FACTOR	8.82	24.55
EFFICIENCY OF THE MAIL-BOX	80 %	83 %

Table 1 : Results

10. Conclusion

We have seen that our algorithm decreases significantly the computation compared to the basic algorithm. But this efficiency is obtained by a subdivision step increasing the memory requirement. On the one hand, this method solves the access to geometrical information problem by a local search in an irregular subdivided space. On the other hand, scene is modeled by a CSG tree which is distributed in the cell structure during the preprocessing phase. Mailboxes contribute efficiently to the gain factor. Future developments about this methods are under consideration, mainly for its application to large scene.

In order to compute large scenes with some thousands of primitives and shortest time, it is necessary to use parallel computer where each processor has local memory. Then we can distribute the data structure of cells among these processors. An implementation of our algorithm on an hypercube iPSC-D5 with 4.5 megabytes of memory for each processor is under progress.

Acknowledgments

We are grateful to Christian Bouville and Roger Brusq who gave some critical remarks of our first implementation using lists. These remarks helped us to improve our algorithm. Thanks also go to Patrice Quinton and Gerard Hegron who provided critical reviews of this paper.

This work has been supported by the CCETT* under contract no 86CN08.

* Centre Commun d'Etudes de Télédiffusion et Télécommunications.

Figure 15 : Interior of a house.

Figure 16 : Jagged ball.

References

[Bouatouch 86] Bouatouch, K., Arnaldi, B., and Priol, T. "LGRC : A language for image synthesis by ray-casting," *TSI* , 6, (Nov-Dec 86), 475-489.

[Blinn 77] Blinn, J.F., "Models of light reflection for computer synthesized pictures," *Computer Graphics* ,11,2 (July 1977), 192-198.

[Bronsvoort 84] Bronsvoort, F., Van Wijk, J.J., and Jansen, F.W., "Two method for improving the efficiency of ray casting in solid modelling," *CAD* ,16,1 (Jan 84), 51-55.

[Cook 82] Cook, R.L., and Torrance, K.E., "A reflectance model for computer graphics," *ACM transactions on graphics*, 1, 1 (Jan 82), 7-24.

[Coquillart 85] Coquillart, S. "An improvement of the ray tracing algorithm," *Eurographics '85 Conference Proceeding*, (1985), 77-88.

[Foley 82] Foley, J.D., and Van Dam, A., *Fundamentals of Interactive Computer Graphics* , Addison, Wesley Publishing Company, London,1982.

[Fuchs 80] Fuchs, H., "On visible surface generation by a priori tree structure," *SIGGRAPH 80 Conference proceeding*, (July 1980) 149-158.

[Glassner 84] Glassner, A., "Space subdivision for fast ray tracing," *IEEE Computer Graphics and Applications*, 4, 10, (Oct 1984),15-22.

[Goldstein 71] Goldstein, R.A., and Nagel, R., "3D modeling with Synthavision simulation," *Simulation*,16, 1, (Jan1971), 25-31.

[Goldstein 79] Goldstein, R.A., and Malin, L., "3D modeling with the Synthavision system," *Proc. First Ann. Conf. Computer Graphics*, CAD/CAM System, (April 1979), 244-247.

[Hall 83] Hall, Roy A., and Greenberg, Donald P., "A testbed for realistic image synthesys," *IEEE Computer Graphics and Applications* , 3, 8, (Nov 1983), 10-20.

[Kajiya 83] Kajiya, J.T., "New techniques for ray tracing procedurally defined objects," *ACM Computer Graphics*, 17 (3), (July1983), 91-102.

[Kaplan 85] Kaplan, M.R. , "Space-tracing, a constant time ray tracer,"*SIGGRAPH 85 tutorial on the uses of spatial coherence in ray tracing.*

[Mantyla 83] Mantyla, M., and Tamminen, M., "Localized set operations for solid modeling," *ACM Computer Graphics*, 17 (3), (July 1983), 279-288.

[Newman 79] Newman, W.M., and Sproul, R.F., *Principle of Interactive Computer Graphics*, 2nd Ed., Mc GRAW HILL, NY,1979.

[Ousterhout 84] Ousterhout, J.K. , "Corner stitching : a data-structuring technique for VLSI layout tools ," *IEEE transactions on CAD* ,3,1, (Jan 1984), 87-100.

[Phong 75] Phong, B.T., " Illumination model for computer generated images," *Communications of the ACM*, 18, (June 1975), 311-317.

[Requicha 80] Requicha, A.A., "Representation for rigid solids : theory, methods, and systems," *ACM Computing Surveys*, 12, 4, (Dec 1980), 437-464.

[Roth 82] Roth, S.D., "Ray casting for modeling solids," *Computer Graphics and Image Processing*, 18, 2, (Feb 1982), 109-144.

[Tamminen 84] Tamminen, M., Koronen, O. , Mantyla, M., "Ray-casting and block model conversion using a spatial index," *CAD* ,16 ,4,(July 1984), 203-208.

[Torrance 67] Torrance, K.E., and Sparrow, E.M., "Theory for off-specular reflection from roughened surfaces," *Journal of Optical Society of America*, 57, 9, (Sept 1967), 1105-1114.

[Whitted 80] Whitted, T., "An improved illumination model for shaded display," *Communications of the ACM*, 23, (June 1980), 343-349.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique